



# MSL Mission Planning & Execution

Presenter:

*Tony Barrett*

Designers/implementers:

*Steve Chien*

*Russell Knight*

*Dan Dvorak*

*Richard Morris*

*Erann Gat*

*Robert Rasmussen*

*Kim Gostelow*

*Thomas Starbird*

*Robert Keller*



# Motivation

- A framework for planning, scheduling, and execution.
- A control centered approach toward representing plans as opposed to an action centered approach
- This is the current default system for flying onboard the MSL rover.



# Outline

- Plan/problem representation
  - Partially ordered constraints on state variables.
- MPE component architecture
  - Numerous threads of execution
- Elaboration & Scheduling
  - Subsumes both PO and HTN planning
- Plan execution
  - Firing time-points & enforcing constraints

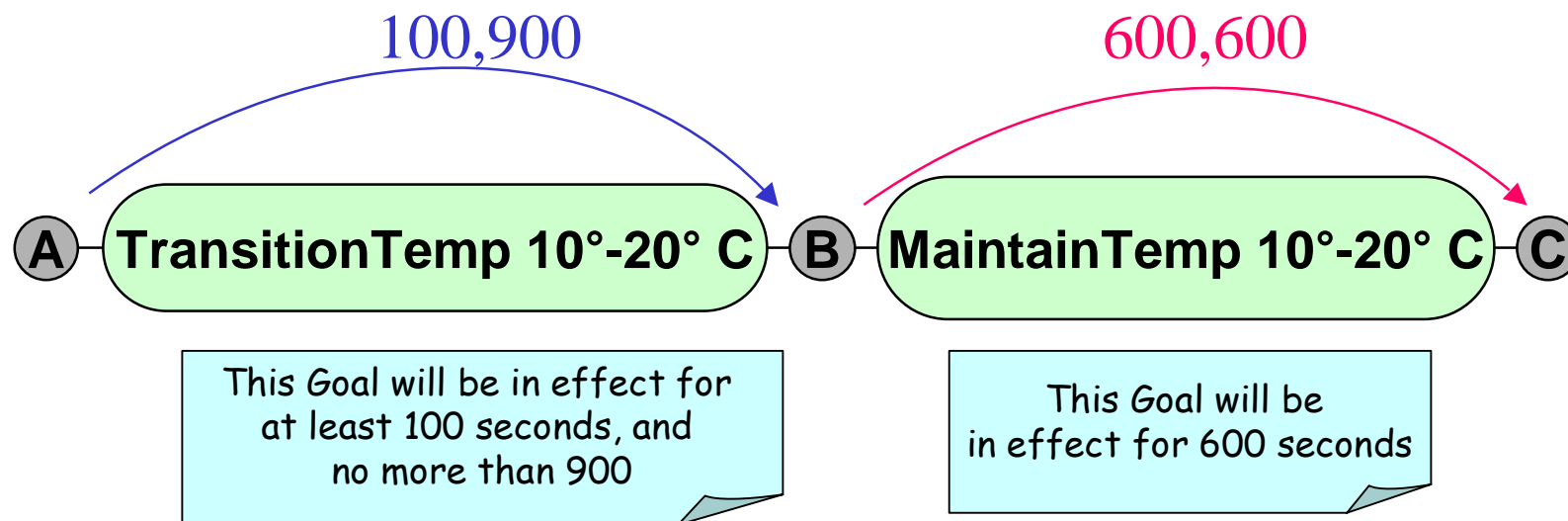


# The Problem

- Givens
  - A model of how state variables affect each other
  - Tactics for elaborating constraints into supporting constraints.
  - A temporally constrained set of commanded constraints on state variables.
- Objectives
  - Elaborate the constraints into an executable network, where constraints can be incrementally passed to controllers for each state variable.
  - Execute the elaborated network.

# Plan/Problem Representation

- A network of timepoints connected by temporal and state-variable constraints.





# Plan/Problem Representation

- Definitions



- Timepoint

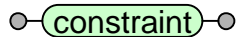
- A flexible point in time



- Temporal Constraint

- A timing relation between two Timepoints

- Goal



- A State Constraint over a time interval demarcated by two Timepoints

- State Constraint

TransitionTemp  
10° -20° C

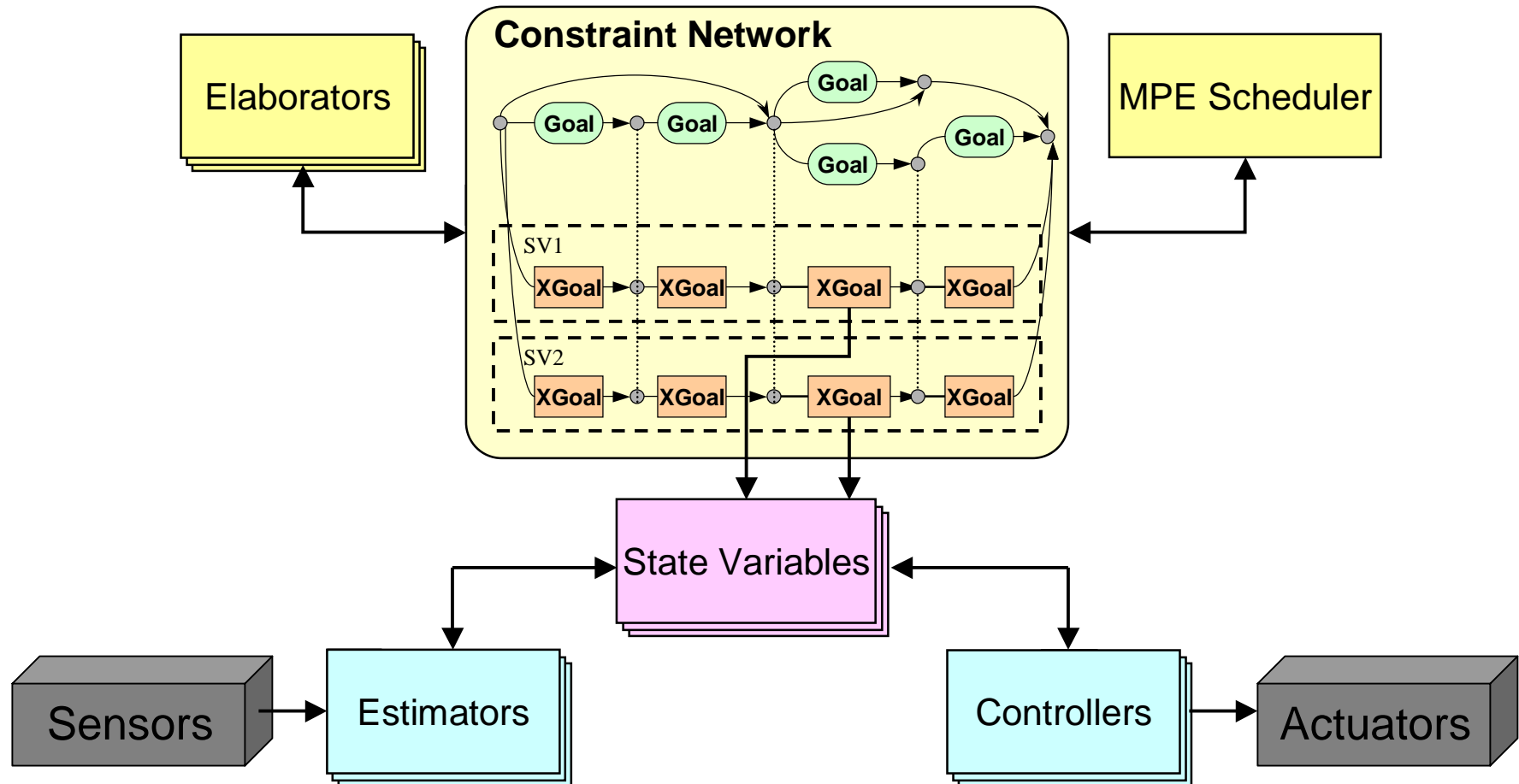
- A representation of a set of values, used to specify a required (allowable) state

- Executable goals (xgoal)

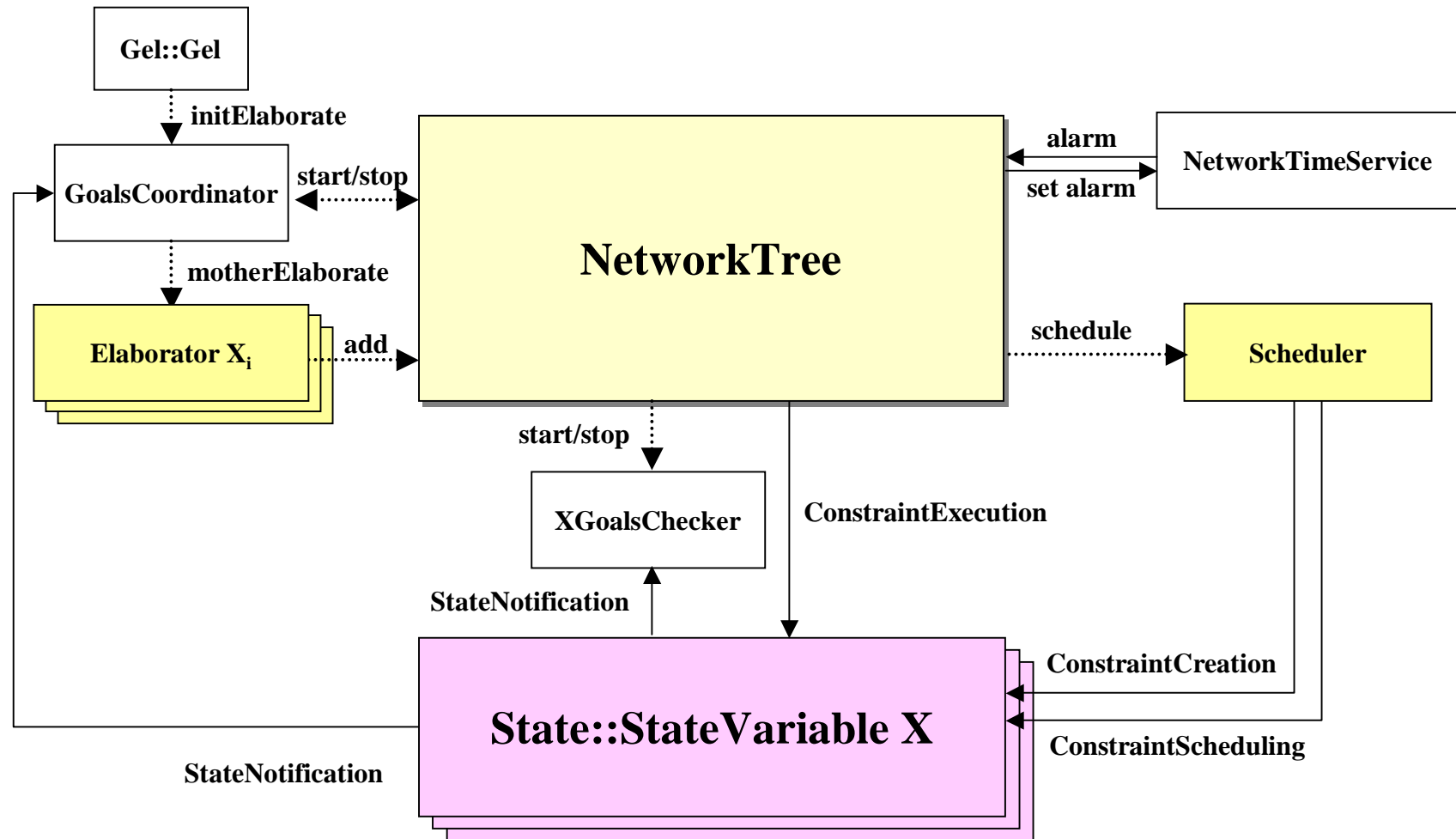


- A merged state constraint that can be passed to a state variable controller

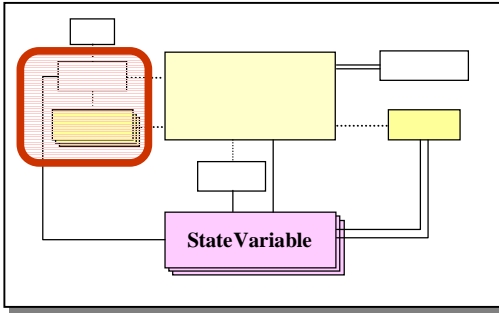
# MPE Component Architecture



# Actual MPE Components (each has an execution thread)







# Elaboration

- Objective

- To set up conditions where a state variable's controller can enforce a goal's constraint.

- Algorithm

Copy task network to proposed network for modification

While there are goals to elaborate do

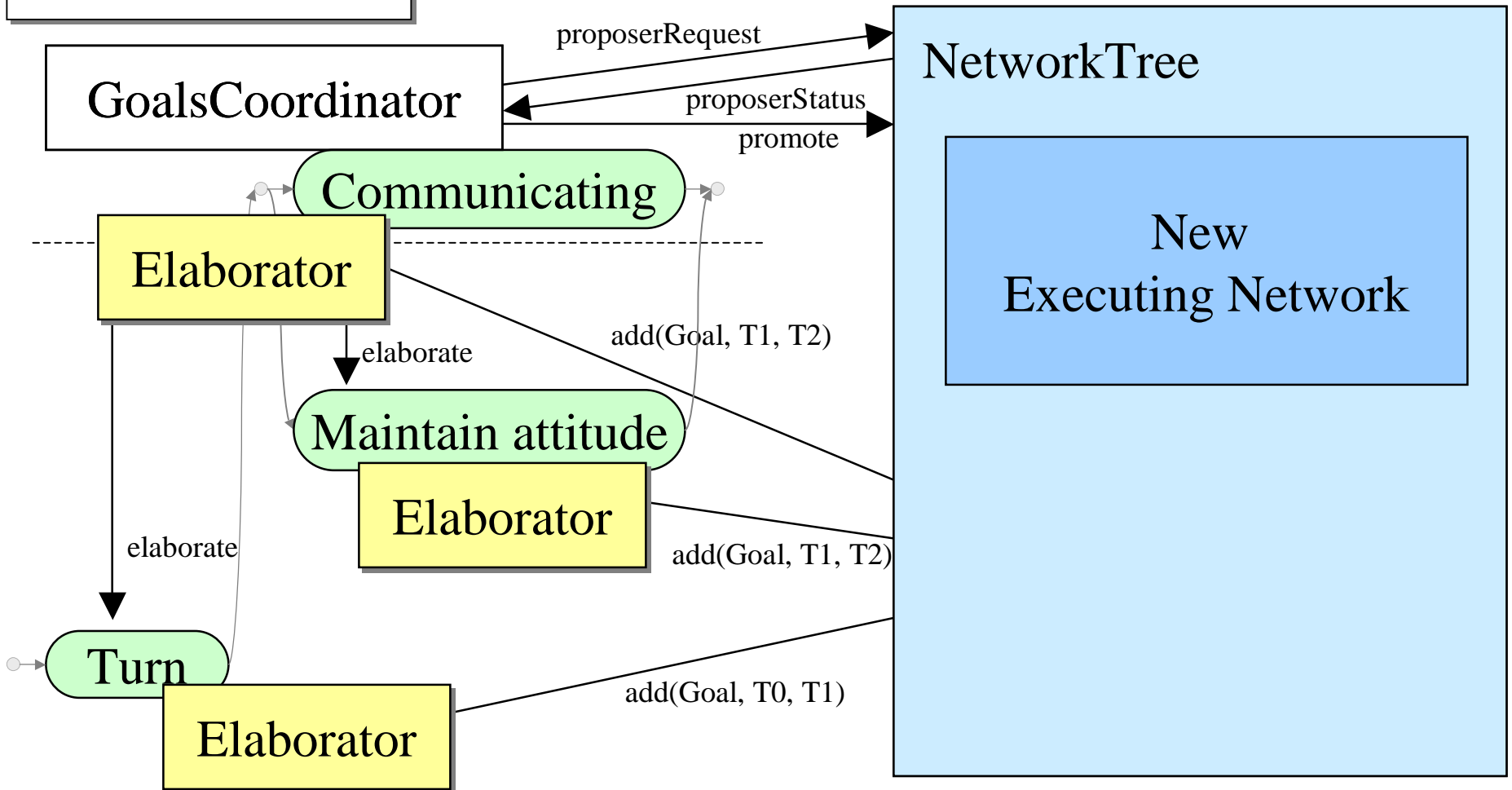
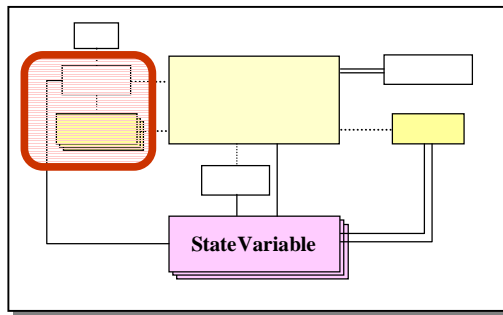
    Choose a goal **G** to elaborate (from a heuristic ordering)

    Exhaustively choose elaboration tactic **E** for **G**

    Apply **E**, which possibly generates new goals to elaborate

        – Backtrack if application of **E** illegal

# Elaboration





# Defining Elaborators

- Default elaborators with GEL

```
(defGoal MyGoal (args)
  (between begin end)

  (tactic MyFirstTactic

    (goal MySubGoal1 (args)
      (between nameTP1 beginTP) )
    (timeConstraint nameTP1 beginTP
      10 20)

    (goal MySubGoal2 (args)
      (between beginTP endTP) )
    (timeConstraint beginTP endTP 5 15)

  ) // MyFirstTactic
) // MyGoal

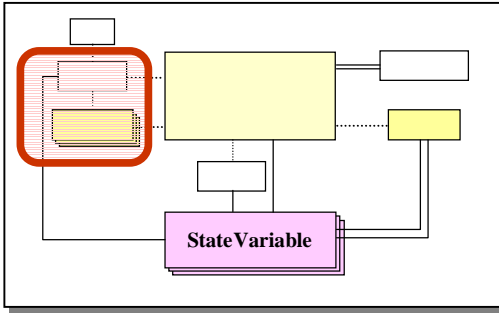
(def tp1 (mkTimepoint tp1))
(def tp2 (mkTimepoint tp2))
(elaborate (mkGoal MyGoal (args)
  (between tp1 tp2)))
```

- Custom elaborators with C++

```
class MyFirstTactic : public Tactic
{
  ElaborationSpec* expand(TimePoint* beginTP,
    TimePoint* endTP)
  {
    addGoal(new MySubGoal1(),
      "nameTP1", beginTP);
    addTemporalConstraint(10, 20, "nameTP1",
      beginTP);

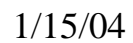
    addGoal(new MySubGoal2(),
      beginTP, endTP);
    addTemporalConstraint(5, 15,
      beginTP, endTP);
  }
};

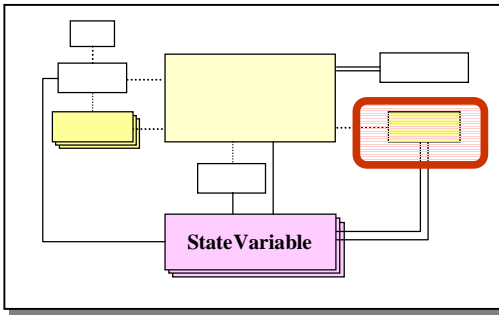
class MyGoal : public Goal<BasicElaborator>
{
  Goal()
  {
    addTactic(new MyFirstTactic() );
    addTactic(new MySecondTactic() );
  }
};
```



# Backtracking

- When an elaborator fails to elaborate
  - Scheduling/propagation fails
  - Cannot find a Tactic that works
- Reports failure to its parent
- Parent can tell a sibling of the failed Elaborator to re-elaborate
  - Doing things a different way could clear up conflicts with the sibling's elaboration
- Then tell the failed Elaborator to “try again”
- Can do this for all SubGoals





# Goal Net Scheduling

- Objective

- To merge goals into executable x-goals

- Algorithm (used during promotion)

For each state variable **SVAR** (from a heuristic ordering) do

For each new goal **G** on **SVAR** (from a heuristic ordering) do

Exhaustively choose **G**'s constrained start timepoint **S**

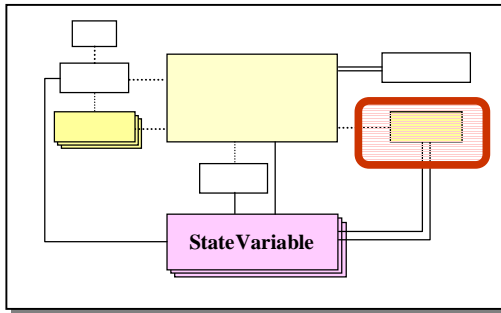
Exhaustively choose **G**'s constrained end timepoint **E**

Merge **G@[S,E]** into **SVAR**'s timeline – Backtrack if merge illegal

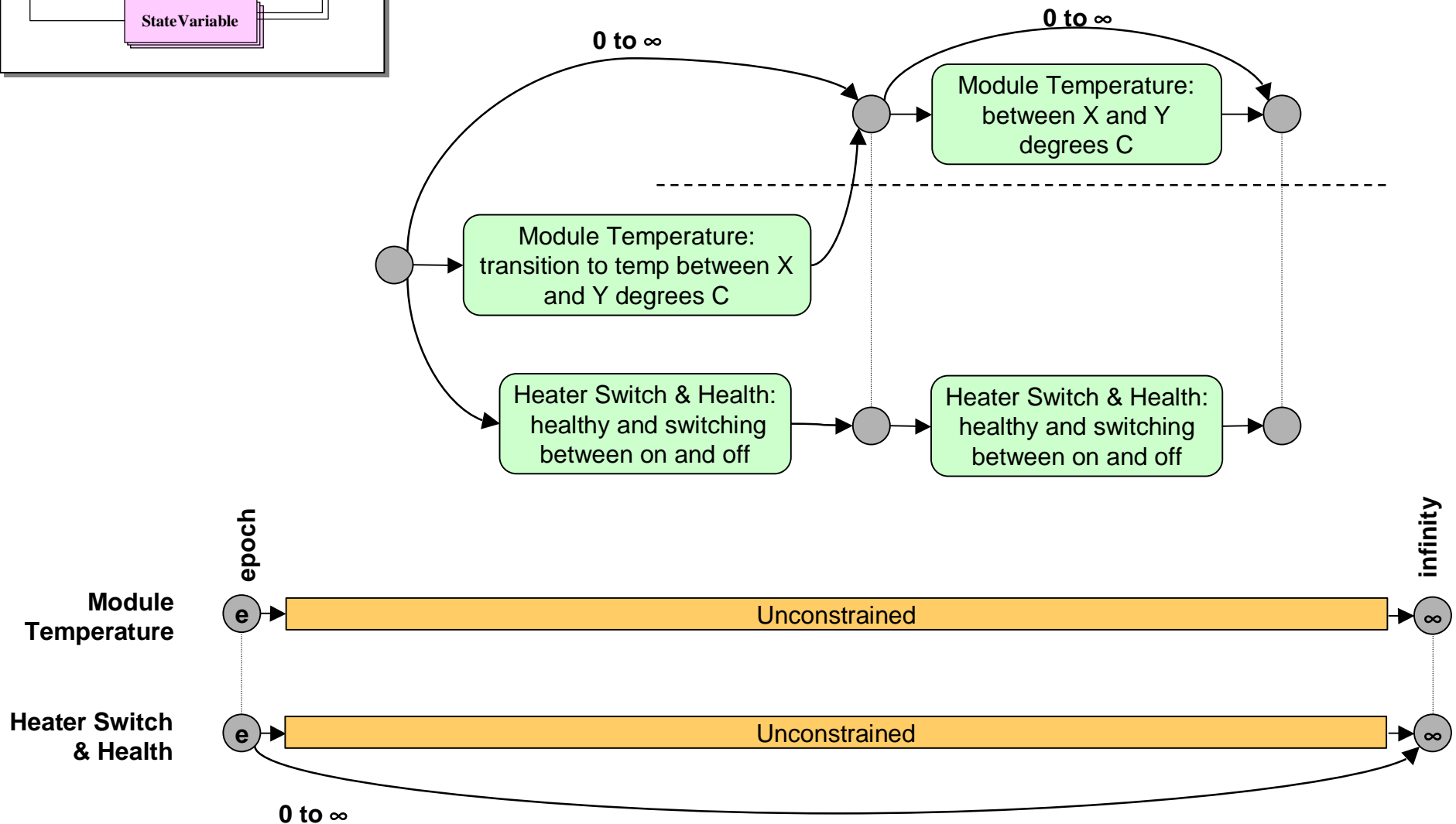
Propagate the expected behavior of **SVAR** given the goals

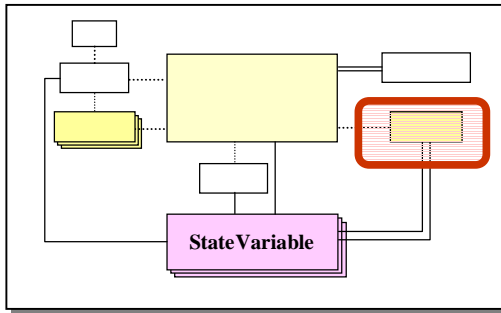
– Backtrack if the propagation illegal

*Replace executing task network with proposed one if it scheduled*

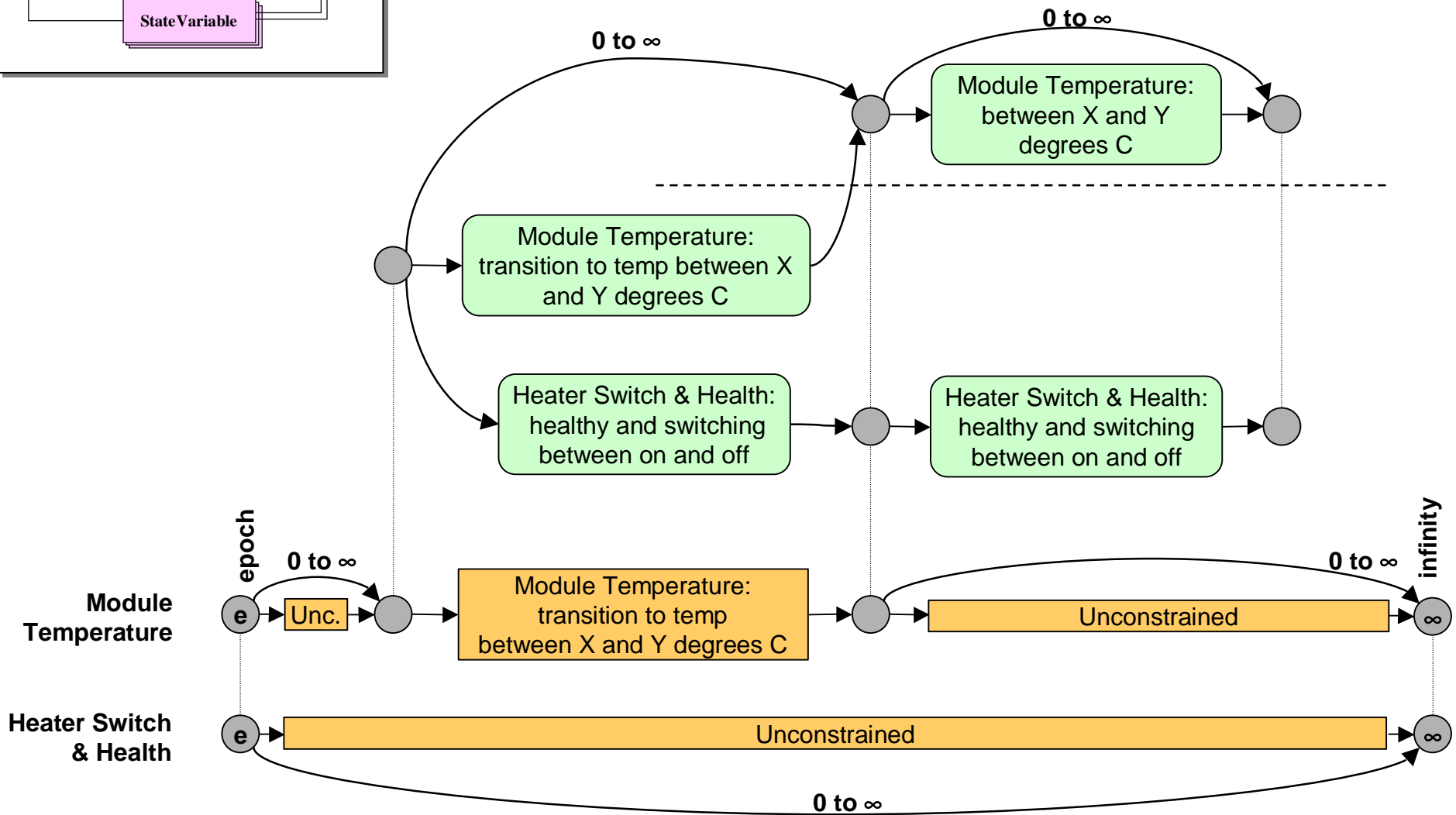


# Computing X-Goals





# Computing X-Goals





The diagram illustrates a state transition system for three modules. States are represented by grey circles, and transitions are labeled with module names and conditions. A dashed line separates the top two modules from the bottom one. The bottom module has an 'Unc.' (uncertain) transition to an infinity state. A thick orange bar at the bottom is labeled 'Unconstrained'.

- Top Module (Temperature):**
  - Initial state transitions to a state labeled "Module Temperature: transition to temp between X and Y degrees C".
  - This state transitions to a state labeled "Module Temperature: between X and Y degrees C".
  - From this state, a transition labeled "0 to  $\infty$ " leads to a final state.
- Middle Module (Heater Switch & Health):**
  - Initial state transitions to a state labeled "Heater Switch & Health: healthy and switching between on and off".
  - This state transitions to a state labeled "Heater Switch & Health: healthy and switching between on and off".
  - From this state, a transition labeled "0 to  $\infty$ " leads to a final state.
- Bottom Module (Temperature):**
  - Initial state transitions to a state labeled "Module Temperature: transition to temp between X and Y degrees C".
  - This state transitions to a state labeled "Module Temperature: between X and Y degrees C".
  - From this state, a transition labeled "Unc." leads to an infinity state.
  - From this state, a transition labeled "0 to  $\infty$ " leads to a final state.

A thick orange bar at the bottom is labeled "Unconstrained".

The diagram illustrates a state transition system for a heater. It is divided into two main sections by a dashed horizontal line. The top section contains three states and three transitions. The bottom section contains three states, two transitions, and a final state labeled 'infinity'. A large pink arrow labeled 'Propagation tests' points from the top section to the bottom section. The bottom section also includes an 'Unconstrained' orange bar and a final state labeled 'infinity'.

**Top Section:**

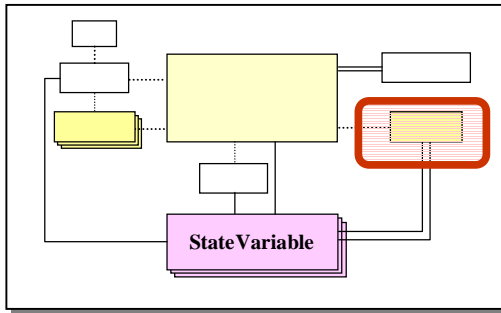
- State 1 (Left):** Initial state.
- Transition 1:** From State 1 to State 2, labeled "Module Temperature: transition to temp between X and Y degrees C" (green box).
- Transition 2:** From State 1 to State 3, labeled "Heater Switch & Health: healthy and switching between on and off" (green box).
- Transition 3:** From State 2 to State 3, labeled "Module Temperature: between X and Y degrees C" (green box).
- Label:** "0 to  $\infty$ " is placed near the transition from State 1 to State 2.

**Bottom Section:**

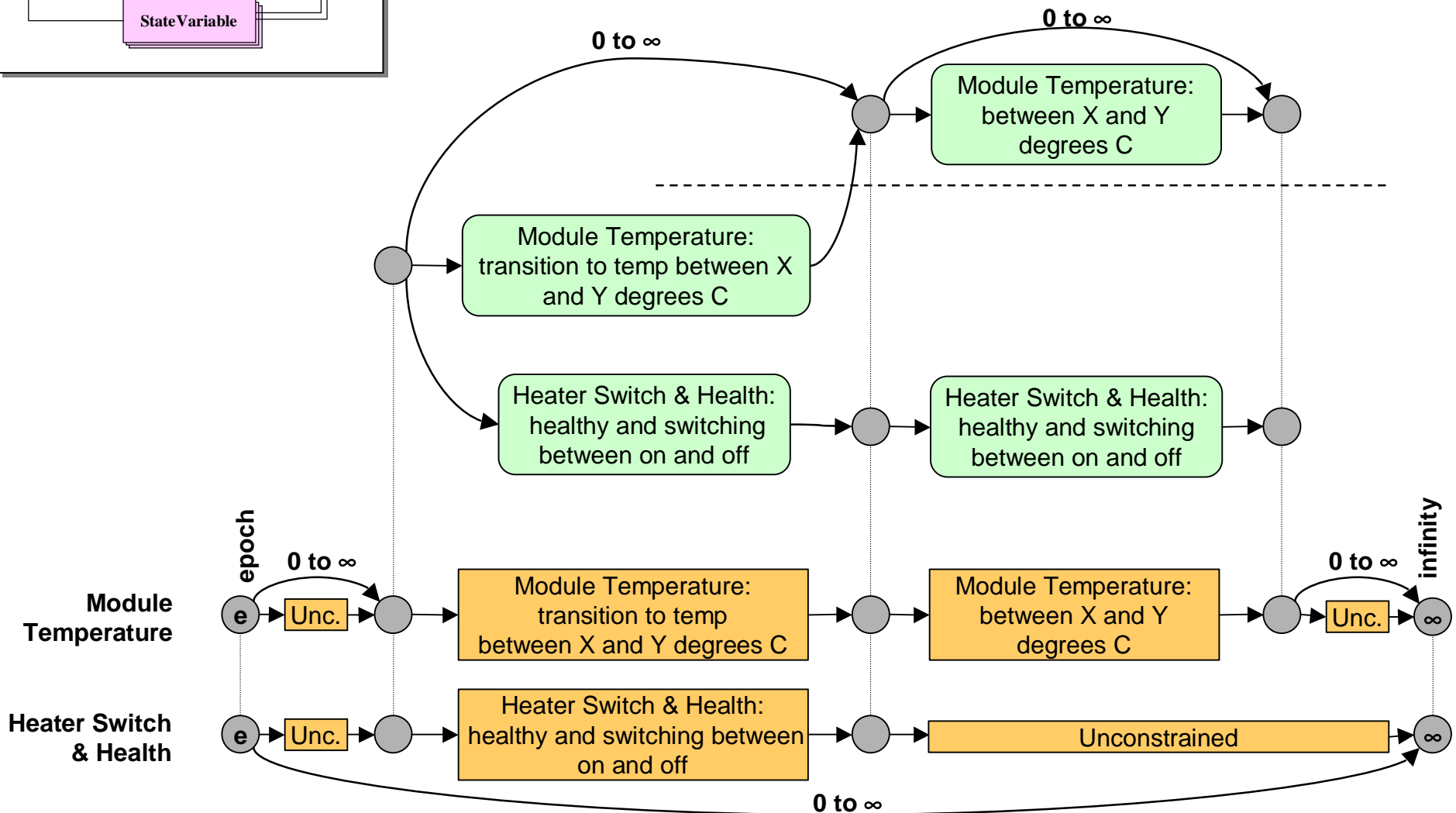
- State 4 (Left):** Initial state.
- Transition 4:** From State 4 to State 5, labeled "Module Temperature: transition to temp between X and Y degrees C" (orange box).
- Transition 5:** From State 5 to State 6, labeled "Module Temperature: between X and Y degrees C" (orange box).
- Transition 6:** From State 6 to State 7, labeled "Unc." (orange box).
- State 7:** Labeled "infinity".
- Label:** "0 to  $\infty$ " is placed near the transition from State 6 to State 7.
- Label:** "Unconstrained" is placed below the transition from State 4 to State 5.
- Label:** "0 to  $\infty$ " is placed below the transition from State 4 to State 7.

**Propagation Tests:**

- A large pink arrow points from the top section to the bottom section, labeled "Propagation tests".
- Three curved arrows point from the top section to the bottom section, indicating the propagation of tests.



# Computing X-Goals



The diagram illustrates a state transition model for a system with two modules. It is divided into two main sections by a horizontal dashed line: the top section represents the 'Healthy' state, and the bottom section represents the 'Uncertain' state.

**Healthy State (Top):**

- Initial State:** A grey circle on the left.
- Transitions:**
  - From the initial state, a curved arrow labeled  $0 \text{ to } \infty$  leads to a grey circle.
  - From this grey circle, a curved arrow labeled  $0 \text{ to } \infty$  leads to another grey circle.
  - From the second grey circle, a curved arrow labeled  $0 \text{ to } \infty$  leads back to the first grey circle.
- Modules (Green Boxes):**
  - Module Temperature:** transition to temp between X and Y degrees C (connected to the first grey circle).
  - Module Temperature:** between X and Y degrees C (connected to the second grey circle).
  - Heater Switch & Health:** healthy and switching between on and off (connected to the first grey circle).
  - Heater Switch & Health:** healthy and switching between on and off (connected to the second grey circle).

**Uncertain State (Bottom):**

- Initial State:** A grey circle on the left.
- Transitions:**
  - From the initial state, a curved arrow labeled  $0 \text{ to } \infty$  leads to a grey circle.
  - From this grey circle, a curved arrow labeled  $0 \text{ to } \infty$  leads to another grey circle.
  - From the second grey circle, a curved arrow labeled  $0 \text{ to } \infty$  leads back to the first grey circle.
- Modules (Orange Boxes):**
  - Module Temperature:** transition to temp between X and Y degrees C (connected to the first grey circle).
  - Module Temperature:** between X and Y degrees C (connected to the second grey circle).
  - Heater Switch & Health:** healthy and switching between on and off (connected to the first grey circle).
  - Heater Switch & Health:** healthy and switching between on and off (connected to the second grey circle).

**Uncertainty and Infinity:**

- On the right side, there are two grey circles labeled  $\infty$ .
- Arrows labeled **Unc.** (Uncertain) point from the grey circles in the 'Uncertain' state to the  $\infty$  circles.
- Curved arrows labeled  $0 \text{ to } \infty$  connect the  $\infty$  circles to each other.
- A curved arrow labeled  $0 \text{ to } \infty$  connects the initial state of the 'Uncertain' state to the  $\infty$  circles.

The diagram illustrates the state transitions for a system with two modules. The states are represented by grey circles, and the transitions are labeled with the action and its frequency (0 to infinity).

**Top Section (Good State - Green Boxes):**

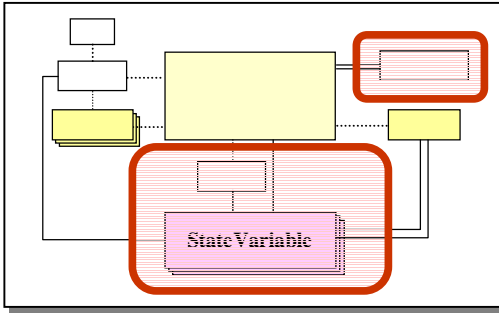
- Module Temperature: transition to temp between X and Y degrees C** (Green box)
- Module Temperature: between X and Y degrees C** (Green box)
- Heater Switch & Health: healthy and switching between on and off** (Green box)

**Bottom Section (Bad State - Orange Boxes):**

- Module Temperature: transition to temp between X and Y degrees C** (Orange box)
- Module Temperature: between X and Y degrees C** (Orange box)
- Heater Switch & Health: healthy and switching between on and off** (Orange box)

**Transitions and Labels:**

- 0 to  $\infty$ :** Transitions from the initial state to the Module Temperature transition box in both sections.
- 0 to  $\infty$ :** Transitions from the Module Temperature transition box to the Module Temperature box in both sections.
- 0 to  $\infty$ :** Transitions from the Module Temperature box to the Heater Switch & Health box in both sections.
- 0 to  $\infty$ :** Transitions from the Heater Switch & Health box to the next Heater Switch & Health box in both sections.
- 0 to  $\infty$ :** Transitions from the Heater Switch & Health box to the infinity state (represented by a circle with an infinity symbol) in both sections.
- Unc.:** Transitions from the Module Temperature box to the infinity state in both sections.
- infinity:** Transitions from the infinity state to the next infinity state in both sections.



# Goal Net Execution

- Objective

- To feed x-goal constraints to state variable controllers as temporal constraints require and circumstances permit.

- Algorithm

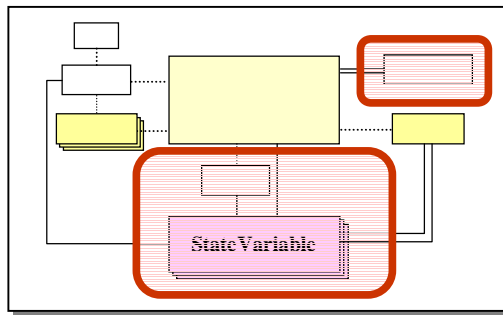
For each unfired timepoint **TP** not temporally constrained into the future do

    If **TP**'s x-goals are ready to start or **TP** is about to time out then (fire)

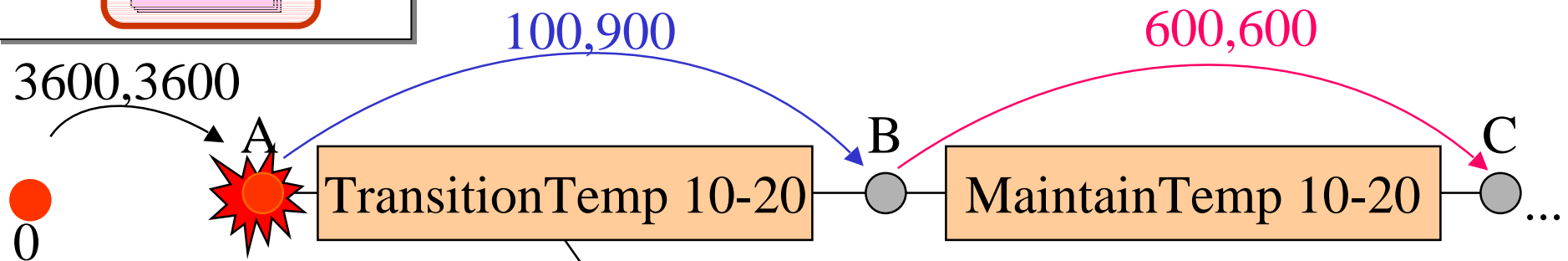
        For each x-goal **G**@[**TP**,\*] respectively do

            Send "start[**G**]" to **G**'s state variable's controller & estimator

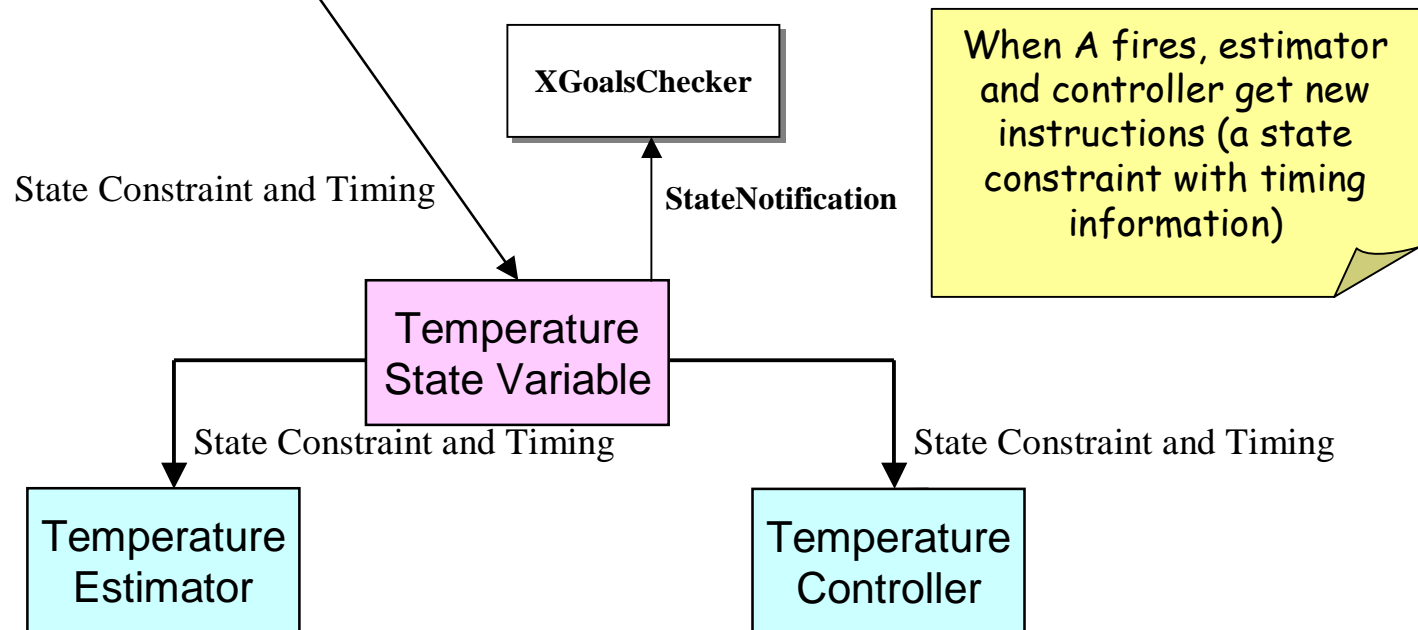
            Inform **G**'s parent goals to start or stop monitoring **G**'s status

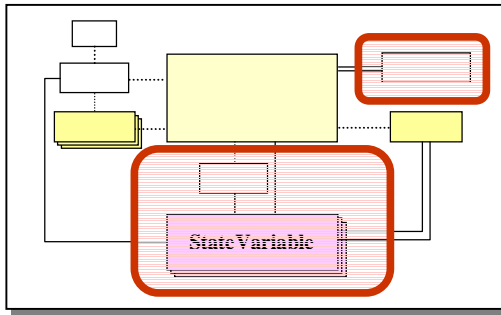


# Timepoint A Fires

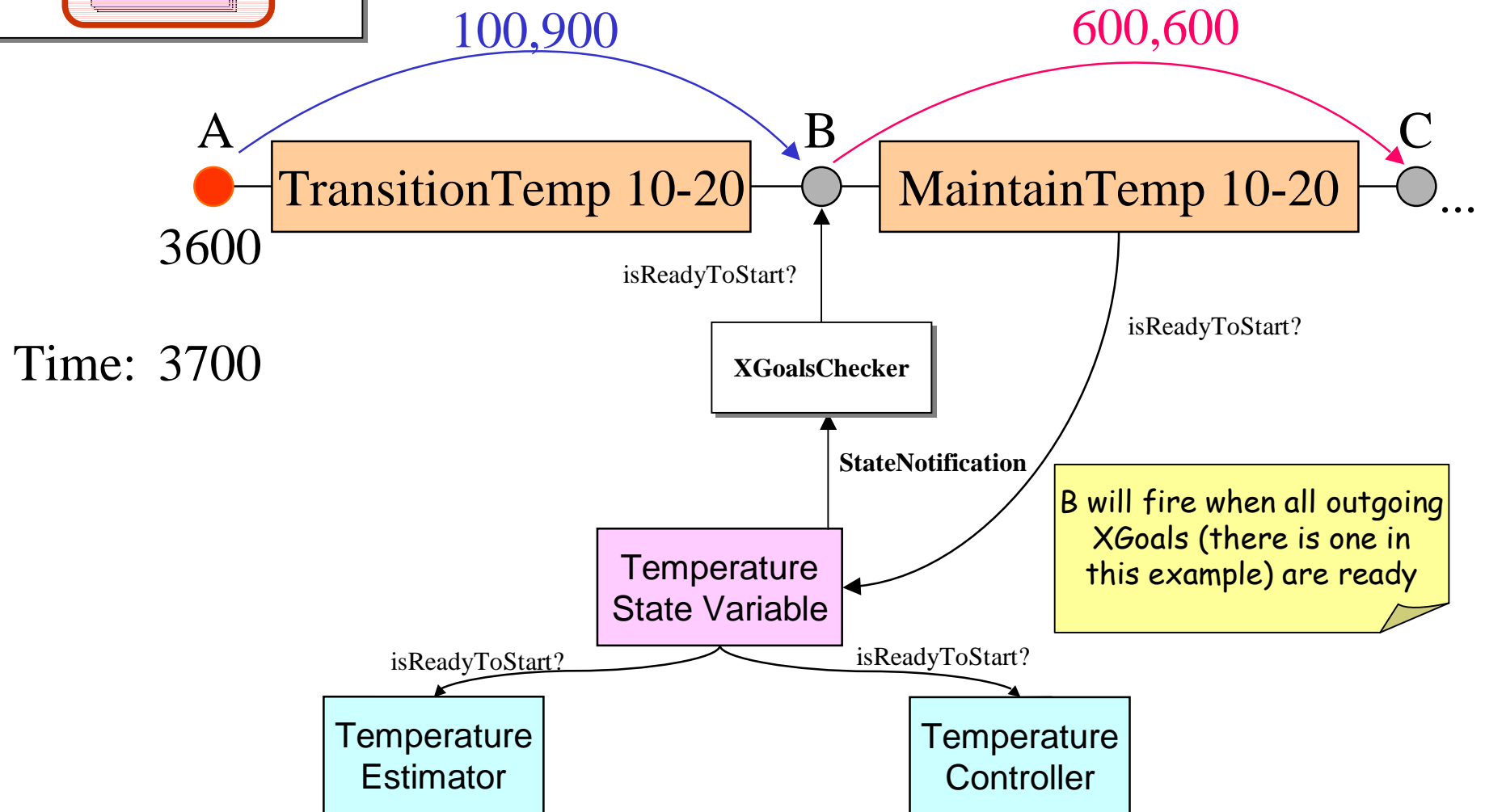


Time: 3600

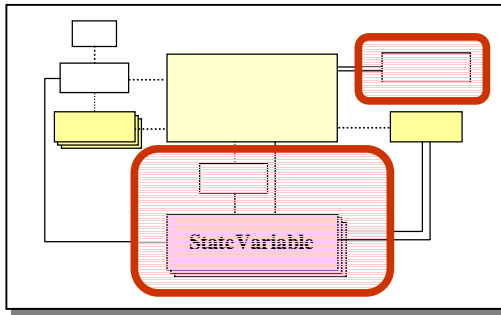




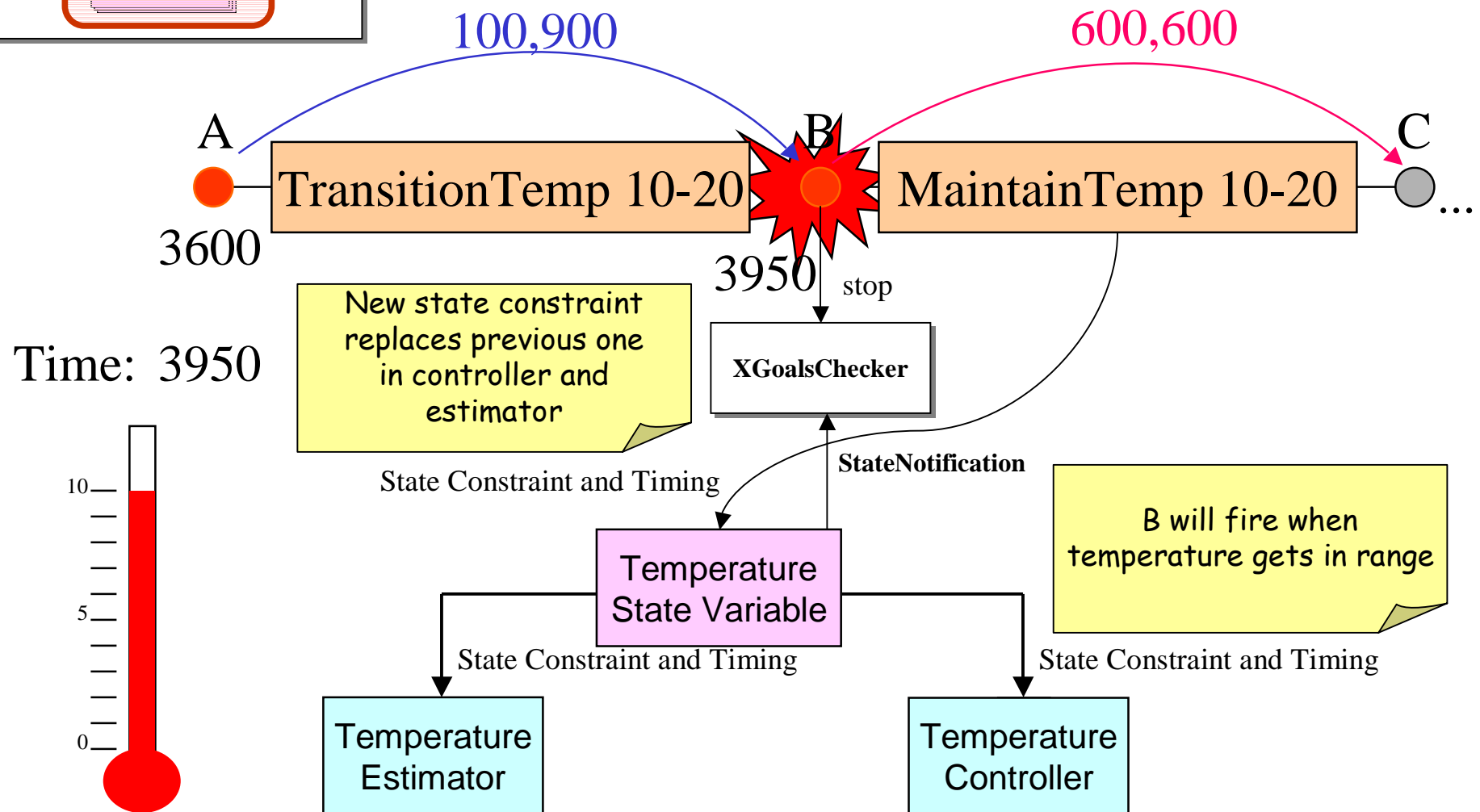
# 100 Seconds Later, Start Conditions are Monitored



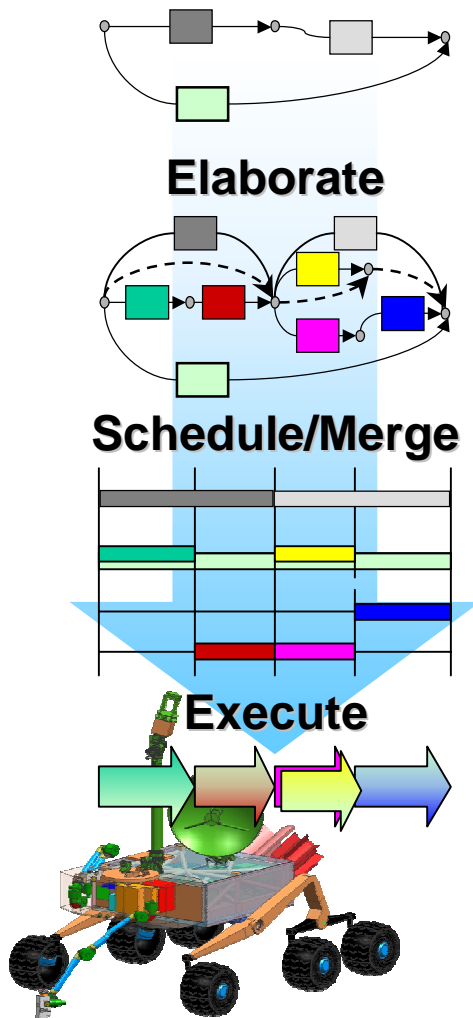




# Timepoint B Fires



# Putting It All Together: a high-level characterization of current MPE algorithm



- **Elaboration algorithm**

Copy task network to proposed network for modification

While there are goals to elaborate do

Choose a goal **G** to elaborate (from a heuristic/random ordering)

Exhaustively choose elaboration tactic **E** for **G**

Apply **E**, which possibly generates new goals to elaborate

– Backtrack if application of **E** illegal

- **Scheduling algorithm (used during promotion)**

For each state variable **SVAR** (from a heuristic/random ordering) do

For each goal **G** on **SVAR** (from a heuristic/random ordering) do

Exhaustively choose **G**'s constrained start timepoint **S**

Exhaustively choose **G**'s constrained end timepoint **E**

Merge **G**@[**S**,**E**] into **SVAR**'s timeline – Backtrack if merge illegal

Propagate the expected behavior of **SVAR** given the goals

– Backtrack if propagation illegal

Replace executing task network with proposed one if it scheduled

- **Execution algorithm**

For each unfired timepoint **TP** not temporally constrained into the future do

If **TP**'s Xgoals are ready to start or **TP** is about to time out then (fire)

For each Xgoal **G**@[**TP**,\*] respectively do

Send "start[**G**]" to **G**'s state variable's controller & estimator

Inform **G**'s parent goals to start or stop monitoring **G**'s status



# Next Steps in the Core

- Prioritized and delayed scheduling
  - Currently elaboration/scheduling just succeeds or fails.
  - Plan to break up commanded task subnet into a set of smaller prioritized subnets to elaborate and schedule in priority order. Subnets that fail to schedule persist for subsequent elaboration/scheduling attempts.
- State affects reasoning
  - Currently the side effects of commanded/elaborated constraints are not modeled.
  - Plan to embed a state-effects model into the system and use it to account for side effects during scheduling.



# Observations

- The focus is on controlling to enforce constraints on state variables.
  - As opposed to the operator focus in AI planning.
  - Facilitates a clear way to merge concurrent action.
- The use of components and elaboration tasks enables a surprising amount of flexibility.
  - GEL can be used, but other approaches to elaboration are possible (in the same implementation).
  - The scheduler component can easily be replaced.
- Lifted time is the default, but it is not necessary.



# References

- See:

<http://mds.jpl.nasa.gov/outreach/>